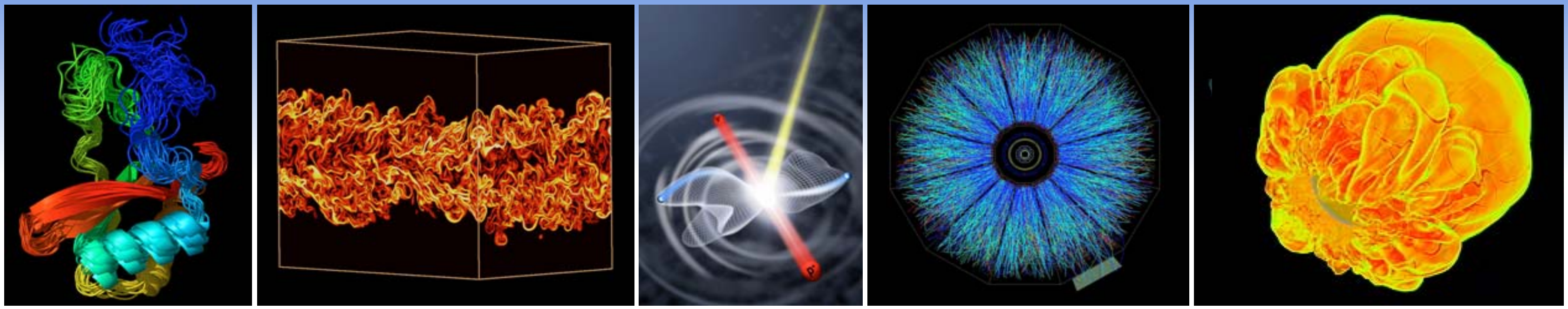


Computing Sciences at Berkeley Lab

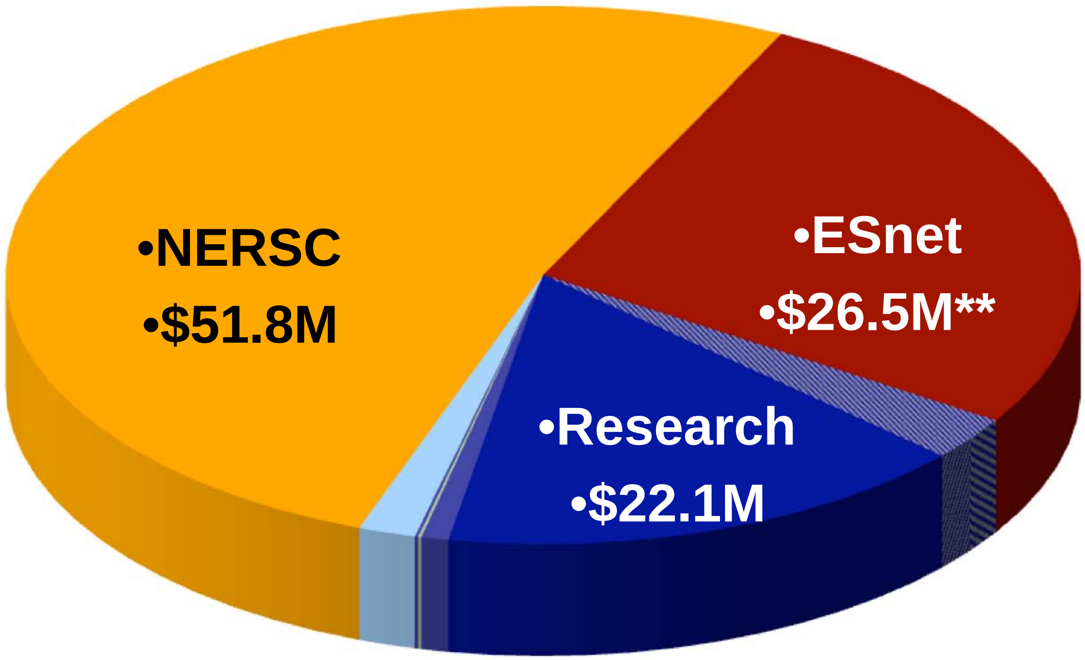
Katherine Yelick
NERSC Director

Computing Sciences Mission

- **Deliver world class facilities, NERSC and ESnet, supporting the DOE Office of Science computational mission**
- **Conduct world class research in applied mathematics and computer science in support of DOE science mission**
- **Build and maintain an outstanding computational science and engineering (CSE) research effort in close collaboration with the UC campuses**

Computing Science Programs

$100.3M Annual Budget in FY09*



•*$3M deducted from FY09 NERSC budget for funds received in FY08

•** ESnet includes 3.2M for services provided to other sites through IWOs

COMPUTING SCIENCES DIRECTORATE

HORST SIMON
Associate Laboratory Director for Computing Sciences

CS DEPUTY
Michael Banda

- BUSINESS MANAGEMENT
- COMMUNICATIONS
- FACILITIES AND EH&S
- HUMAN RESOURCES

NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING (NERSC) DIVISION

KATHY YELICK
Division Director

HOWARD WALTER
Division Deputy

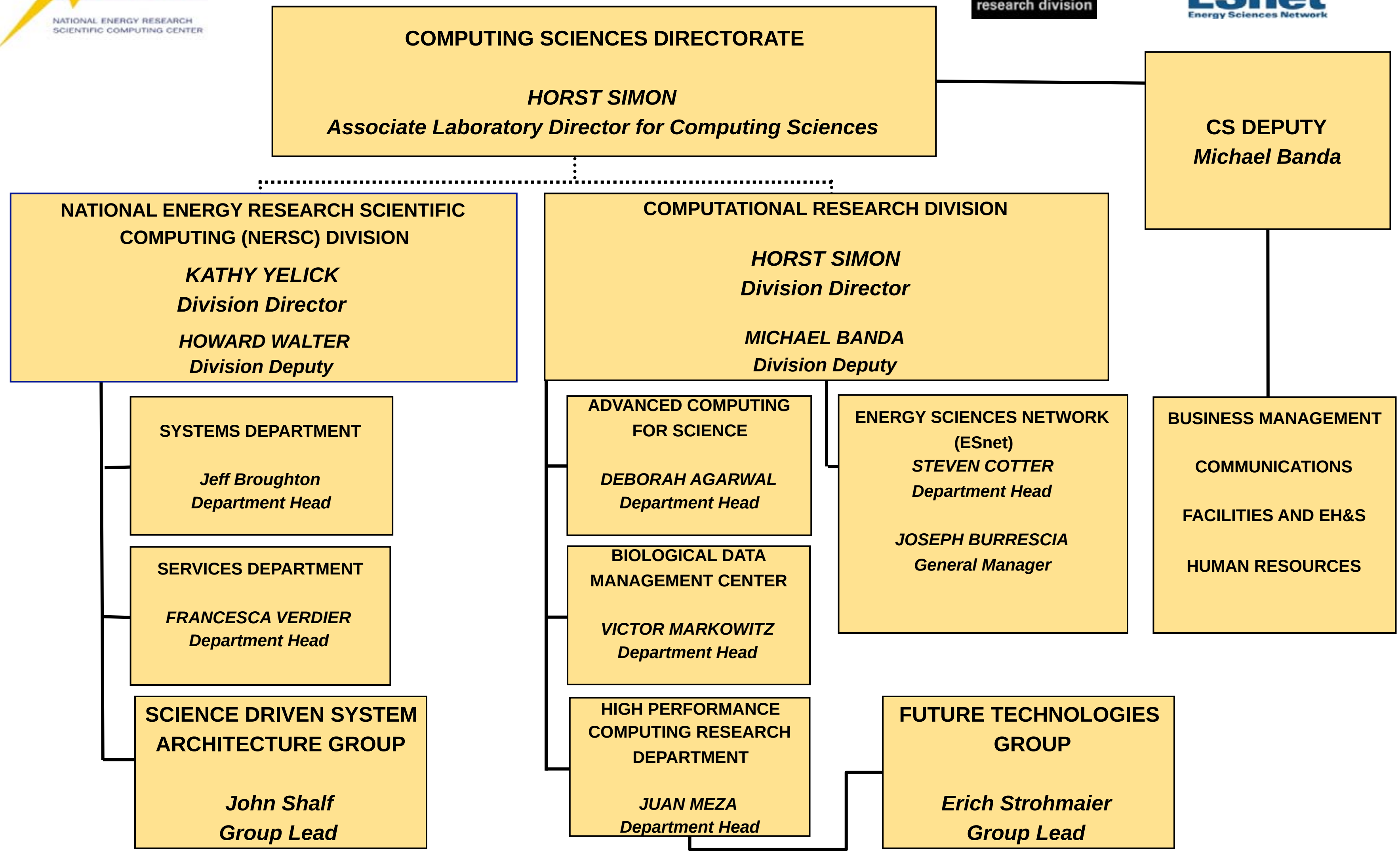
- SYSTEMS DEPARTMENT
  Jeff Broughton
  Department Head
- SERVICES DEPARTMENT
  FRANCESCA VERDIER
  Department Head

COMPUTATIONAL RESEARCH DIVISION

HORST SIMON
Division Director

MICHAEL BANDA
Division Deputy

- ADVANCED COMPUTING FOR SCIENCE
  DEBORAH AGARWAL
  Department Head
- BIOLOGICAL DATA MANAGEMENT CENTER
  VICTOR MARKOWITZ
  Department Head
- HIGH PERFORMANCE COMPUTING RESEARCH DEPARTMENT
  JUAN MEZA
  Department Head
- ENERGY SCIENCES NETWORK (ESnet)
  STEVEN COTTER
  Department Head
  JOSEPH BURRESCIA
  General Manager



COMPUTING SCIENCES DIRECTORATE
HORST SIMON
Associate Laboratory Director for Computing Sciences
CS DEPUTY
Michael Banda
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING (NERSC) DIVISION
KATHY YELICK
Division Director
HOWARD WALTER
Division Deputy
COMPUTATIONAL RESEARCH DIVISION
HORST SIMON
Division Director
MICHAEL BANDA
Division Deputy
SYSTEMS DEPARTMENT
Jeff Broughton
Department Head
SERVICES DEPARTMENT
FRANCESCA VERDIER
Department Head
SCIENCE DRIVEN SYSTEM ARCHITECTURE GROUP
John Shalf
Group Lead
ADVANCED COMPUTING FOR SCIENCE
DEBORAH AGARWAL
Department Head
BIOLOGICAL DATA MANAGEMENT CENTER
VICTOR MARKOWITZ
Department Head
HIGH PERFORMANCE COMPUTING RESEARCH DEPARTMENT
JUAN MEZA
Department Head
ENERGY SCIENCES NETWORK (ESnet)
STEVEN COTTER
Department Head
JOSEPH BURRESCIA
General Manager
BUSINESS MANAGEMENT
COMMUNICATIONS
FACILITIES AND EH&S
HUMAN RESOURCES
FUTURE TECHNOLOGIES GROUP
Erich Strohmaier
Group Lead

Overview of the Berkeley UPC Project

UPC Project Goals

2001-2004: A Portable UPC Compiler

- UPC was (incorrectly) viewed as a language that required shared memory hardware or only ran on Cray machines
- The Berkeley UPC compiler showed it could run on clusters with a lightweight runtime and that source-to-source translation was reasonable

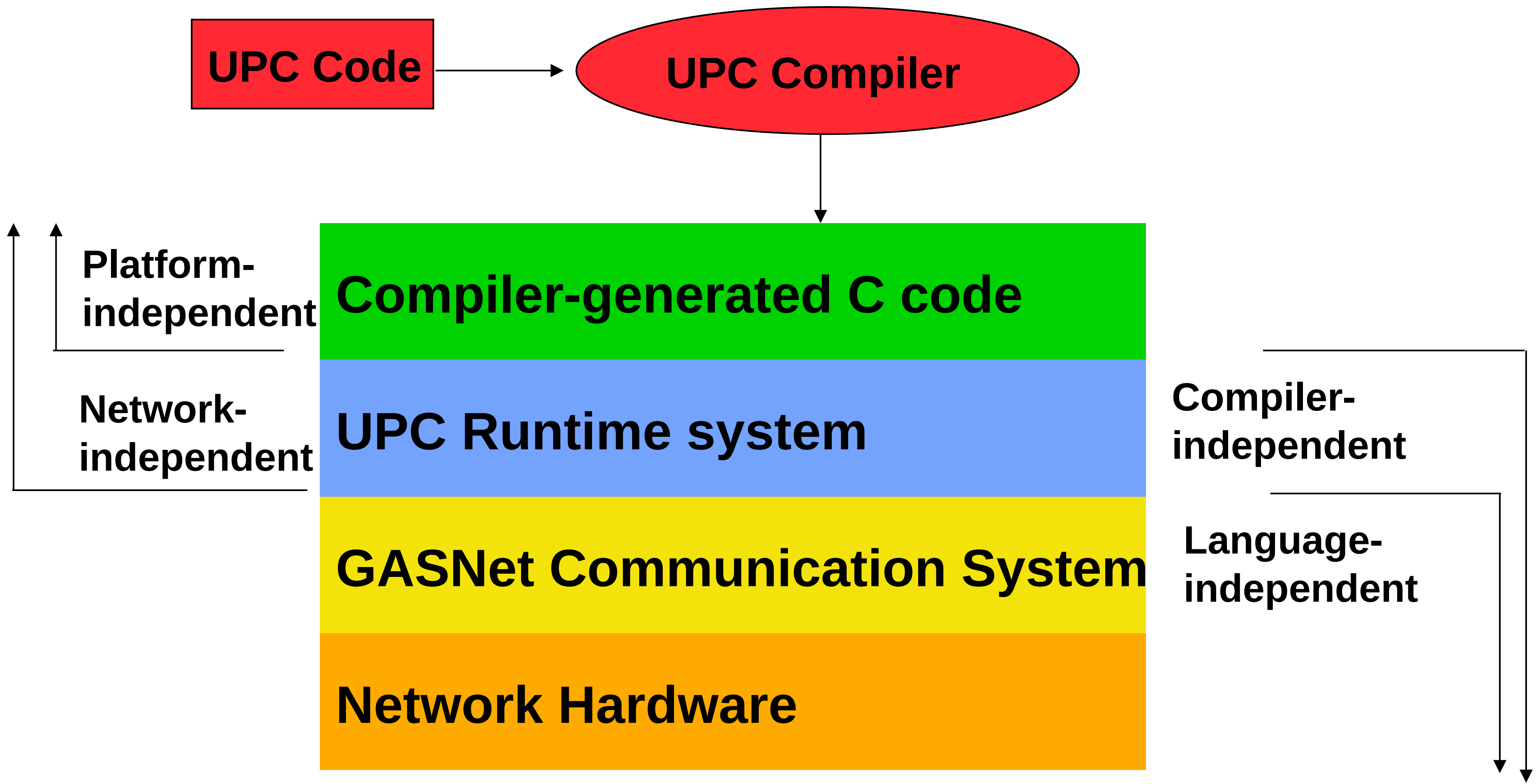
2005-2008: UPC is a High Performance Language

- Conventional wisdom: UPC is more productive than MPI but we should expect it to be slower (maybe by 2x)
- Even on clusters without global address space support, UPC can outperform MPI on microbenchmarks and apps

2008-2010: UPC for multicore & hybrid multicore / clusters

- Are either MPI / core or OpenMP within a node “good enough”?
- UPC should be better: lower memory footprint, better locality contol

Berkeley UPC Compiler



Berkeley UPC Compiler Highlights

- Portable, high-performance open-source UPC compiler
  - Fully UPC spec compliant
  - Includes UPC collectives and UPC-I/O
- Many extensions for performance and programmability
  - Non-blocking and non-contiguous memcpy functions
  - Semaphores and signaling put
  - Fine granularity timers
  - Value-based collectives
  - Atomic memory operations
  - Hierarchical layout query
  - Call to/from MPI (C++, F, etc.)
- Entirely free & open source
  - Available from http://upc.lbl.gov
  - Also in CDs at SC each year

New in November 2008 Release

- Native support for IBM BlueGene/P (dcmf conduit)
- Cray Portals network upgrade to cache local memory registration
- Many small improvements to IBM LAPI support
- Improvements in Myrinet GM support
- InfiniBand support for InfiniPath adapters
- Experimental support for ARM processors
- MIPSEL/Linux (SiCortex) support (alas)
- Fine-grained programs optimizations
  - split-phase, access coalescing
- Beta release of “vectorization”
- Pathscale Compiler support
- Many bug fixes

Multicore (and Accelerator) Plans

- **Performance is often non-obvious**
  - E.g., MPI faster than threads on an SMPs
- **Multiple runtime approaches**
  - SMP runtime system based on Pthreads (old)
    - Some work to pin threads and get memory affinity for NUMA
    - Some overhead for thread-local (globally scoped) data
  - Run a GASNet conduit with processes
    - High overhead for communication, but good data partitioning
  - Processes with some form of OS-supported shared memory
    - Prototype done by Jason Duell, but only for a single node
    - More from Filip today
- **Partitioned memory space (GPUs, Cell, etc.)**
  - Some work on this by Filip for Cell
  - More ideas from Yili and rest of the group

Collective Communication

- **Important for many scientific applications**
  - Productivity and performance enhancer
  - Teams are critical: more on this later by Yili
- **Collectives for clusters**
  - Work with IBM, data centric
  - New work: completely re-designed and re-built collectives
  - Some improvements in Nov 08 release, more coming
- **Collectives for Multicore**
  - Surprisingly important, even for barriers
- **Autotuned collectives (Yili and Rajesh)**
  - Taking the pain out of tuning

GASNet

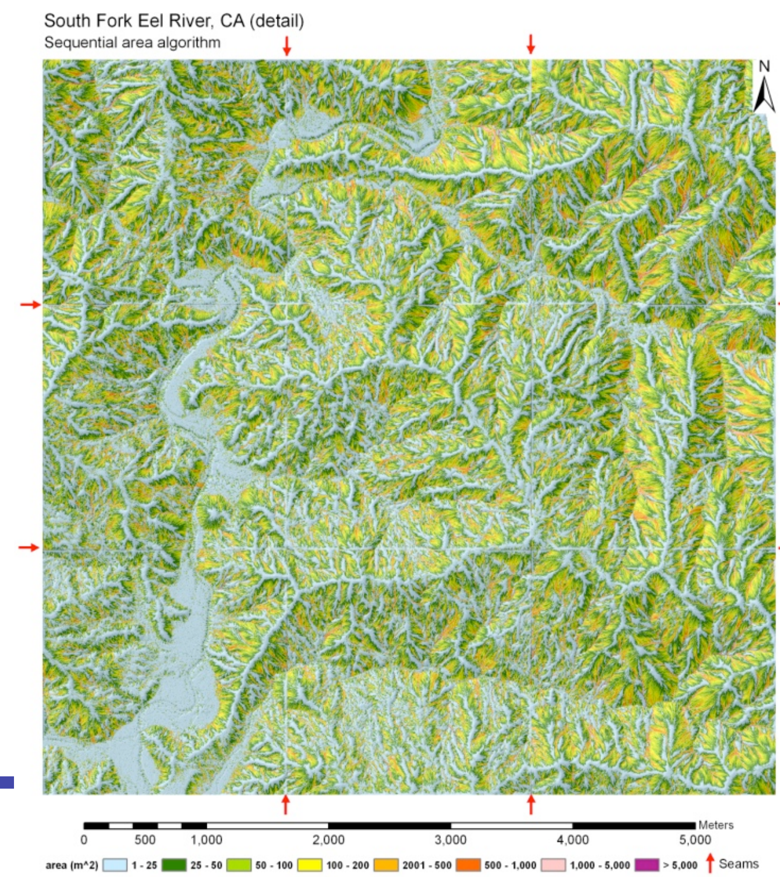
- **Focus on BG/P conduit (Rajesh)**
  - Scaling work on Intrepid machine at Argonne (ALCF)
- **Cray Portals conduit (Paul)**
  - Several bug fixes and performance improvements
  - Ongoing benchmarking work on Franklin machine at NERSC and others
- **Infiniband**
  - Ongoing bug fixed and performance improvements
  - Work specifically on scaling for Ranger machine at TACC

UPC Applications from LBNL

- **UPC applications stress global address space:**
  - irregular remote memory accesses
  - need for low overhead communication
- **Delaunay mesh generation**
- **Adaptive Mesh Refinement**
- **Sparse Cholesky factorization**
- **Biology application**
- **Dense LU factorization with event-driven execution**
- **Latest one: Landscape evolution (NSF PetaApps Climate project)**

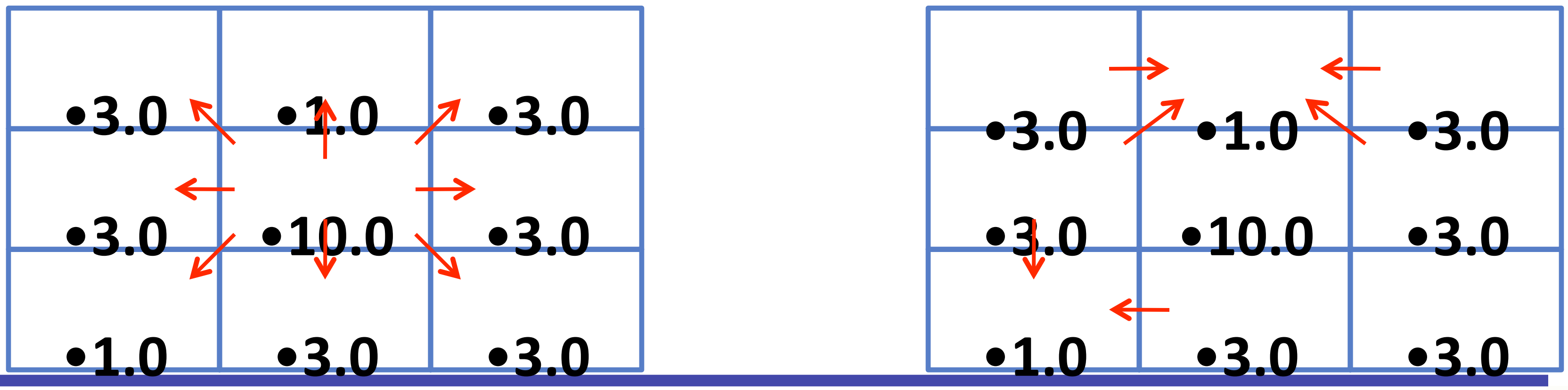
Landscape Evolution

- **How to evolve landscapes over time?**
  - Erosion, rainfall, river incision
- **Area distribution**
  - Want to parallelize this
- **Input series of tiles**
  - Vast input range
  - Serial seams



Area Distribution Algorithm

- **Analogous to precipitation distribution**
  - Each cell seeded with an initial area distribution
  - Area "distributed" proportionally to downhill neighbors based on elevation
- **Originally a recursive, serial algorithm**
  - Changed to a scatter operation for parallel implementation

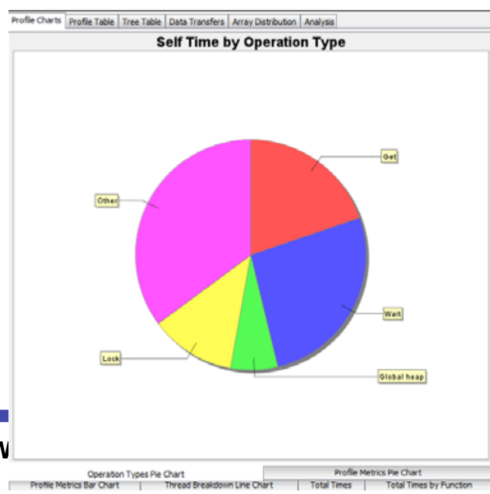


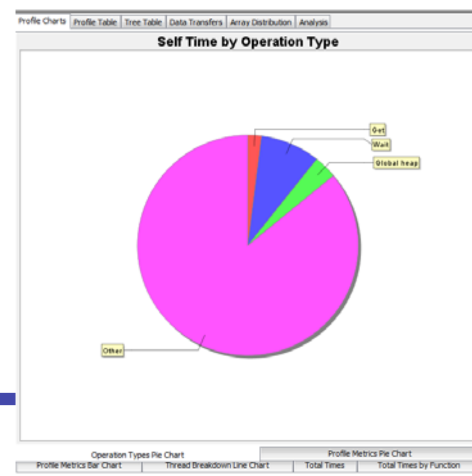
UPC Implementation

- **Input elevation tiles stored in global shared array**
  - Threads split input tiles with horizontal stripes
- **Information shared after collecting all uphill data**
  - Reduces communication across threads
  - Queues used to push/pop information to/from remote threads
- **Threads alternate work between local/remote queues to improve throughput**
- **Computation ceases once all cells have pushed data**

Implementations

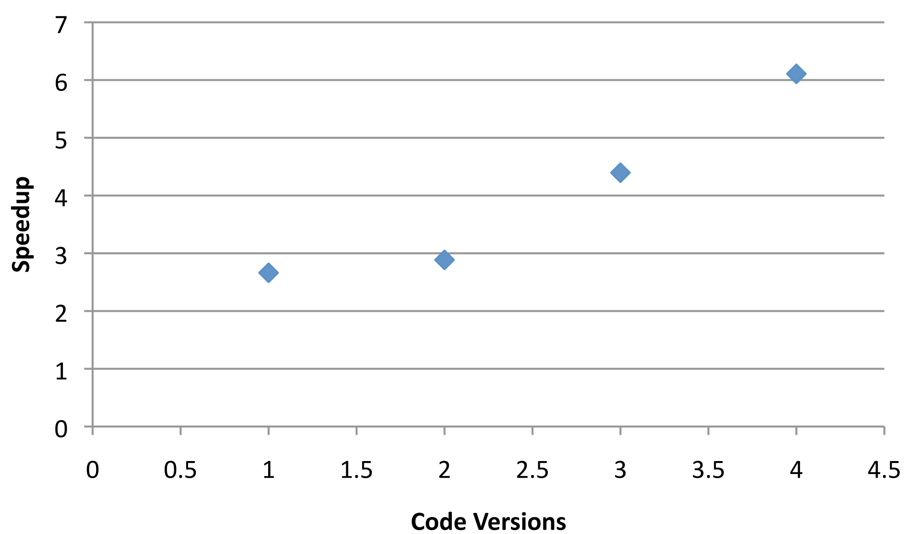
- **Original implementation was very inefficient**
  - Remote variables, upc_locks, non-local queues
- **PPW helpful in locating areas of inefficiency**
  - Indicated areas in source with unexpected communication
  - UPC functions comprised > 60% of runtime





Optimizations

- **V 1 – Optimized parallel base-code**
- **V 2 – Remote queues local to data owner**
- **V 3 – Replaced queues with dynamic arrays for local computation**
- **V 4 – Replaced UPC locks with semaphores**



Tests performed with eight 2km x 2km tiles and eight threads

Analysis

- **Speedup capped by topography**
  - In worst case, performance could be less than serial code
  - In best case, with minimal computation, can achieve a speedup of 7.1 for prior test case
- **Eliminating UPC locks gave best results in terms of scalability**
- **PPW helpful for pinpointing areas for improvement**
- **Running algorithm in parallel is only way to achieve the true results**

9:00 a.m.	Overview of Computing Sciences and Berkeley UPC Project	Kathy Yelick
9:30 a.m.	Berkeley Compiler Update	Costin Iancu
10:00 a.m.	Intrepid Compiler Update	Gary Funck
10:30 a.m.	Break	
10:45 a.m.	Team Collectives and BG/P Results	Yili and Rajesh
11:30	GASNet on Cray Portals	Paul Hargrove
12:00 p.m.	Working Lunch	
	Process-Based SMP Runtime	Filip Blagojevic
1:00 p.m.	Autotuned Multicore Collectives	Rajesh Nishtala
1:30 p.m.	Resource Management for Multicore	Costin Iancu
2:15 p.m	Compiling Shared Memory programs for GPUs	Seung-Jai Min
2:40 p.m.	UPC Ideas for GPUs / Accelerators	Yili Zheng
3:15 p.m.	Break	
3:30 p.m.	Irregular Communication Optimizations in PGAS	Jimmy Su
4:00 p.m.	Irregular Memory Optimizations on Multicore	Kamesh Madduri
4:30 p.m.	Future Plans	Kathy Yelick